[ScrollMagic](http://docs.google.com/index.html)

* [Classes](http://docs.google.com/classes.list.html)
  + [Controller](http://docs.google.com/ScrollMagic.Controller.html)
  + [Scene](http://docs.google.com/ScrollMagic.Scene.html)
* [Events](http://docs.google.com/events.list.html)
  + [add](http://docs.google.com/ScrollMagic.Scene.html#event:add)
  + [change](http://docs.google.com/ScrollMagic.Scene.html#event:change)
  + [destroy](http://docs.google.com/ScrollMagic.Scene.html#event:destroy)
  + [end](http://docs.google.com/ScrollMagic.Scene.html#event:end)
  + [enter](http://docs.google.com/ScrollMagic.Scene.html#event:enter)
  + [leave](http://docs.google.com/ScrollMagic.Scene.html#event:leave)
  + [progress](http://docs.google.com/ScrollMagic.Scene.html#event:progress)
  + [remove](http://docs.google.com/ScrollMagic.Scene.html#event:remove)
  + [shift](http://docs.google.com/ScrollMagic.Scene.html#event:shift)
  + [start](http://docs.google.com/ScrollMagic.Scene.html#event:start)
  + [update](http://docs.google.com/ScrollMagic.Scene.html#event:update)
* [Plugins](http://docs.google.com/mixins.list.html)
  + [GSAP](http://docs.google.com/animation.GSAP.html)
  + [Velocity](http://docs.google.com/animation.Velocity.html)
  + [addIndicators](http://docs.google.com/debug.addIndicators.html)
  + [jQuery](http://docs.google.com/framework.jQuery.html)

[**ScrollMagic**](http://docs.google.com/ScrollMagic.html)**.**Scene

new ScrollMagic.Scene**(options)**

A Scene defines where the controller should react and how.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| options | object | <optional> | Options for the Scene. The options can be updated at any time.  Instead of setting the options for each scene individually you can also set them globally in the controller as the controllers globalSceneOptions option. The object accepts the same properties as the ones below.  When a scene is added to the controller the options defined using the Scene constructor will be overwritten by those set in globalSceneOptions. Properties  | Name | Type | Argument | Default | Description | | --- | --- | --- | --- | --- | | duration | number | string | function | <optional> | 0 | The duration of the scene.  Please see Scene.duration() for details. | | offset | number | <optional> | 0 | Offset Value for the Trigger Position. If no triggerElement is defined this will be the scroll distance from the start of the page, after which the scene will start. | | triggerElement | string | object | <optional> | null | Selector or DOM object that defines the start of the scene. If undefined the scene will start right at the start of the page (unless an offset is set). | | triggerHook | number | string | <optional> | "onCenter" | Can be a number between 0 and 1 defining the position of the trigger Hook in relation to the viewport.  Can also be defined using a string:   * "onEnter" => 1 * "onCenter" => 0.5 * "onLeave" => 0 | | reverse | boolean | <optional> | true | Should the scene reverse, when scrolling up? | | loglevel | number | <optional> | 2 | Loglevel for debugging. Note that logging is disabled in the minified version of ScrollMagic.   * 0 => silent * 1 => errors * 2 => errors, warnings * 3 => errors, warnings, debuginfo | |

Source:

* [ScrollMagic/Scene.js](http://docs.google.com/ScrollMagic_Scene.js.html), [line 39](http://docs.google.com/ScrollMagic_Scene.js.html#sunlight-1-line-39)

##### Example

// create a standard scene and add it to a controller  
new ScrollMagic.Scene()  
 .addTo(controller);  
  
// create a scene with custom options and assign a handler to it.  
var scene = new ScrollMagic.Scene({  
 duration: 100,  
 offset: 200,  
 triggerHook: "onEnter",  
 reverse: false  
});

### Scene Control Methods

#### addTo**(controller) → {Scene}**

Add the scene to a controller.

This is the equivalent to Controller.addScene(scene).

##### Parameters:

| Name | Type | Description |
| --- | --- | --- |
| controller | [ScrollMagic.Controller](http://docs.google.com/ScrollMagic.Controller.html) | The controller to which the scene should be added. |

Source:

* [ScrollMagic/Scene/core.js](http://docs.google.com/ScrollMagic_Scene_core.js.html), [line 34](http://docs.google.com/ScrollMagic_Scene_core.js.html#sunlight-1-line-34)

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

// add a scene to a ScrollMagic Controller  
scene.addTo(controller);

#### controller**() → {**[**ScrollMagic.Controller**](http://docs.google.com/ScrollMagic.Controller.html)**}**

**Get** the associated controller.

Source:

* [ScrollMagic/Scene/getters-setters.js](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html), [line 235](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html#sunlight-1-line-235)

##### Returns:

Parent controller or undefined

{ [ScrollMagic.Controller](http://docs.google.com/ScrollMagic.Controller.html) }

##### Example

// get the controller of a scene  
var controller = scene.controller();

#### destroy**(reset) → {null}**

Destroy the scene and everything.

##### Parameters:

| Name | Type | Argument | Default | Description |
| --- | --- | --- | --- | --- |
| reset | boolean | <optional> | false | If true the pin and tween (if existent) will be reset. |

Source:

* [ScrollMagic/Scene/core.js](http://docs.google.com/ScrollMagic_Scene_core.js.html), [line 117](http://docs.google.com/ScrollMagic_Scene_core.js.html#sunlight-1-line-117)

##### Returns:

Null to unset handler variables.

{ null }

##### Example

// destroy the scene without resetting the pin and tween to their initial positions  
scene = scene.destroy();  
  
// destroy the scene and reset the pin and tween  
scene = scene.destroy(true);

#### progress**(progress) → {number}**

**Get** or **Set** the scene's progress.

Usually it shouldn't be necessary to use this as a setter, as it is set automatically by scene.update().

The order in which the events are fired depends on the duration of the scene:

1. Scenes with duration == 0:  
   Scenes that have no duration by definition have no ending. Thus the end event will never be fired.  
   When the trigger position of the scene is passed the events are always fired in this order:  
   enter, start, progress when scrolling forward  
   and  
   progress, start, leave when scrolling in reverse
2. Scenes with duration > 0:  
   Scenes with a set duration have a defined start and end point.  
   When scrolling past the start position of the scene it will fire these events in this order:  
   enter, start, progress  
   When continuing to scroll and passing the end point it will fire these events:  
   progress, end, leave  
   When reversing through the end point these events are fired:  
   enter, end, progress  
   And when continuing to scroll past the start position in reverse it will fire:  
   progress, start, leave  
   In between start and end the progress event will be called constantly, whenever the progress changes.

In short:

enter events will always trigger **before** the progress update and leave envents will trigger **after** the progress update.

start and end will always trigger at their respective position.

Please review the event descriptions for details on the events and the event object that is passed to the callback.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| progress | number | <optional> | The new progress value of the scene [0-1]. |

Source:

* [ScrollMagic/Scene/core.js](http://docs.google.com/ScrollMagic_Scene_core.js.html), [line 215](http://docs.google.com/ScrollMagic_Scene_core.js.html#sunlight-1-line-215)

##### Fires:

* Scene.enter,event: when used as setter
* Scene.start,event: when used as setter
* Scene.progress,event: when used as setter
* Scene.end,event: when used as setter
* Scene.leave,event: when used as setter

##### Returns:

* get - Current scene progress.{ number }
* set - Parent object for chaining.{ Scene }

##### Example

// get the current scene progress  
var progress = scene.progress();  
  
// set new scene progress  
scene.progress(0.3);

#### refresh**() → {Scene}**

Updates dynamic scene variables like the trigger element position or the duration.

This method is automatically called in regular intervals from the controller. See [ScrollMagic.Controller](http://docs.google.com/ScrollMagic.Controller.html) option refreshInterval.

You can call it to minimize lag, for example when you intentionally change the position of the triggerElement.

If you don't it will simply be updated in the next refresh interval of the container, which is usually sufficient.

Since:

* 1.1.0

Source:

* [ScrollMagic/Scene/core.js](http://docs.google.com/ScrollMagic_Scene_core.js.html), [line 186](http://docs.google.com/ScrollMagic_Scene_core.js.html#sunlight-1-line-186)

##### Fires:

* Scene.shift,event: if the trigger element position or the duration changed
* Scene.change,event: if the duration changed

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

scene = new ScrollMagic.Scene({triggerElement: "#trigger"});  
  
// change the position of the trigger  
$("#trigger").css("top", 500);  
// immediately let the scene know of this change  
scene.refresh();

#### remove**() → {Scene}**

Remove the scene from the controller.

This is the equivalent to Controller.removeScene(scene).

The scene will not be updated anymore until you readd it to a controller.

To remove the pin or the tween you need to call removeTween() or removePin() respectively.

Source:

* [ScrollMagic/Scene/core.js](http://docs.google.com/ScrollMagic_Scene_core.js.html), [line 93](http://docs.google.com/ScrollMagic_Scene_core.js.html#sunlight-1-line-93)

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

// remove the scene from its controller  
scene.remove();

#### removeClassToggle**(reset) → {Scene}**

Remove the class binding from the scene.

##### Parameters:

| Name | Type | Argument | Default | Description |
| --- | --- | --- | --- | --- |
| reset | boolean | <optional> | false | If false and the classes are currently active, they will remain on the element. If true they will be removed. |

Source:

* [ScrollMagic/Scene/feature-classToggles.js](http://docs.google.com/ScrollMagic_Scene_feature-classToggles.js.html), [line 47](http://docs.google.com/ScrollMagic_Scene_feature-classToggles.js.html#sunlight-1-line-47)

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

// remove class binding from the scene without reset  
scene.removeClassToggle();  
  
// remove class binding and remove the changes it caused  
scene.removeClassToggle(true);

#### removePin**(reset) → {Scene}**

Remove the pin from the scene.

##### Parameters:

| Name | Type | Argument | Default | Description |
| --- | --- | --- | --- | --- |
| reset | boolean | <optional> | false | If false the spacer will not be removed and the element's position will not be reset. |

Source:

* [ScrollMagic/Scene/feature-pinning.js](http://docs.google.com/ScrollMagic_Scene_feature-pinning.js.html), [line 342](http://docs.google.com/ScrollMagic_Scene_feature-pinning.js.html#sunlight-1-line-342)

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

// remove the pin from the scene without resetting it (the spacer is not removed)  
scene.removePin();  
  
// remove the pin from the scene and reset the pin element to its initial position (spacer is removed)  
scene.removePin(true);

#### setClassToggle**(element, classes) → {Scene}**

Define a css class modification while the scene is active.

When the scene triggers the classes will be added to the supplied element and removed, when the scene is over.

If the scene duration is 0 the classes will only be removed if the user scrolls back past the start position.

##### Parameters:

| Name | Type | Description |
| --- | --- | --- |
| element | string | object | A Selector targeting one or more elements or a DOM object that is supposed to be modified. |
| classes | string | One or more Classnames (separated by space) that should be added to the element during the scene. |

Source:

* [ScrollMagic/Scene/feature-classToggles.js](http://docs.google.com/ScrollMagic_Scene_feature-classToggles.js.html), [line 9](http://docs.google.com/ScrollMagic_Scene_feature-classToggles.js.html#sunlight-1-line-9)

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

// add the class 'myclass' to the element with the id 'my-elem' for the duration of the scene  
scene.setClassToggle("#my-elem", "myclass");  
  
// add multiple classes to multiple elements defined by the selector '.classChange'  
scene.setClassToggle(".classChange", "class1 class2 class3");

#### setPin**(element, settings) → {Scene}**

Pin an element for the duration of the scene.

If the scene duration is 0 the element will only be unpinned, if the user scrolls back past the start position.

Make sure only one pin is applied to an element at the same time.

An element can be pinned multiple times, but only successively.

***NOTE:*** *The option pushFollowers has no effect, when the scene duration is 0.*

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| element | string | object |  | A Selector targeting an element or a DOM object that is supposed to be pinned. |
| settings | object | <optional> | settings for the pin Properties  | Name | Type | Argument | Default | Description | | --- | --- | --- | --- | --- | | pushFollowers | boolean | <optional> | true | If true following elements will be "pushed" down for the duration of the pin, if false the pinned element will just scroll past them.  Ignored, when duration is 0. | | spacerClass | string | <optional> | "scrollmagic-pin-spacer" | Classname of the pin spacer element, which is used to replace the element. | |

Source:

* [ScrollMagic/Scene/feature-pinning.js](http://docs.google.com/ScrollMagic_Scene_feature-pinning.js.html), [line 180](http://docs.google.com/ScrollMagic_Scene_feature-pinning.js.html#sunlight-1-line-180)

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

// pin element and push all following elements down by the amount of the pin duration.  
scene.setPin("#pin");  
  
// pin element and keeping all following elements in their place. The pinned element will move past them.  
scene.setPin("#pin", {pushFollowers: false});

#### update**(immediately) → {Scene}**

Updates the Scene to reflect the current state.

This is the equivalent to Controller.updateScene(scene, immediately).

The update method calculates the scene's start and end position (based on the trigger element, trigger hook, duration and offset) and checks it against the current scroll position of the container.

It then updates the current scene state accordingly (or does nothing, if the state is already correct) – Pins will be set to their correct position and tweens will be updated to their correct progress.

This means an update doesn't necessarily result in a progress change. The progress event will be fired if the progress has indeed changed between this update and the last.

***NOTE:*** *This method gets called constantly whenever ScrollMagic detects a change. The only application for you is if you change something outside of the realm of ScrollMagic, like moving the trigger or changing tween parameters.*

##### Parameters:

| Name | Type | Argument | Default | Description |
| --- | --- | --- | --- | --- |
| immediately | boolean | <optional> | false | If true the update will be instant, if false it will wait until next update cycle (better performance). |

Source:

* [ScrollMagic/Scene/core.js](http://docs.google.com/ScrollMagic_Scene_core.js.html), [line 139](http://docs.google.com/ScrollMagic_Scene_core.js.html#sunlight-1-line-139)

##### Fires:

* Scene.event:update

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

// update the scene on next tick  
scene.update();  
  
// update the scene immediately  
scene.update(true);

### Scene Parameters (getter / setter)

#### duration**(newDuration) → {number}**

**Get** or **Set** the duration option value.

As a **setter** it accepts three types of parameters:

1. number: Sets the duration of the scene to exactly this amount of pixels.  
   This means the scene will last for exactly this amount of pixels scrolled. Sub-Pixels are also valid.  
   A value of 0 means that the scene is 'open end' and no end will be triggered. Pins will never unpin and animations will play independently of scroll progress.
2. string: Always updates the duration relative to parent scroll container.  
   For example "100%" will keep the duration always exactly at the inner height of the scroll container.  
   When scrolling vertically the width is used for reference respectively.
3. function: The supplied function will be called to return the scene duration.  
   This is useful in setups where the duration depends on other elements who might change size. By supplying a function you can return a value instead of updating potentially multiple scene durations.  
   The scene can be referenced inside the callback using this.  
   ***WARNING:*** *This is an easy way to kill performance, as the callback will be executed every time Scene.refresh() is called, which happens a lot. The interval is defined by the controller (see ScrollMagic.Controller option refreshInterval).*  
   *It's recomended to avoid calculations within the function and use cached variables as return values.*  
   *This counts double if you use the same function for multiple scenes.*

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| newDuration | number | string | function | <optional> | The new duration setting for the scene. |

Source:

* [ScrollMagic/Scene/getters-setters.js](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html), [line 102](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html#sunlight-1-line-102)

##### Fires:

* Scene.change,event: when used as setter
* Scene.shift,event: when used as setter

##### Returns:

* get - Current scene duration.{ number }
* set - Parent object for chaining.{ Scene }

##### Example

// get the current duration value  
var duration = scene.duration();  
  
// set a new duration  
scene.duration(300);  
  
// set duration responsively to container size  
scene.duration("100%");  
  
// use a function to randomize the duration for some reason.  
var durationValueCache;  
function durationCallback () {  
 return durationValueCache;  
}  
function updateDuration () {  
 durationValueCache = Math.random() \* 100;  
}  
updateDuration(); // set to initial value  
scene.duration(durationCallback); // set duration callback

#### enabled**(newState) → {boolean|Scene}**

**Get** or **Set** the current enabled state of the scene.

This can be used to disable this scene without removing or destroying it.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| newState | boolean | <optional> | The new enabled state of the scene true or false. |

Source:

* [ScrollMagic/Scene/core.js](http://docs.google.com/ScrollMagic_Scene_core.js.html), [line 68](http://docs.google.com/ScrollMagic_Scene_core.js.html#sunlight-1-line-68)

##### Returns:

Current enabled state or parent object for chaining.

{ boolean | Scene }

##### Example

// get the current value  
var enabled = scene.enabled();  
  
// disable the scene  
scene.enabled(false);

#### loglevel**(newLoglevel) → {number}**

**Get** or **Set** the loglevel option value.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| newLoglevel | number | <optional> | The new loglevel setting of the scene. [0-3] |

Source:

* [ScrollMagic/Scene/getters-setters.js](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html), [line 219](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html#sunlight-1-line-219)

##### Fires:

* Scene.change,event: when used as setter

##### Returns:

* get - Current loglevel.{ number }
* set - Parent object for chaining.{ Scene }

##### Example

// get the current loglevel  
var loglevel = scene.loglevel();  
  
// set new loglevel  
scene.loglevel(3);

#### offset**(newOffset) → {number}**

**Get** or **Set** the offset option value.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| newOffset | number | <optional> | The new offset of the scene. |

Source:

* [ScrollMagic/Scene/getters-setters.js](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html), [line 148](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html#sunlight-1-line-148)

##### Fires:

* Scene.change,event: when used as setter
* Scene.shift,event: when used as setter

##### Returns:

* get - Current scene offset.{ number }
* set - Parent object for chaining.{ Scene }

##### Example

// get the current offset  
var offset = scene.offset();  
  
// set a new offset  
scene.offset(100);

#### reverse**(newReverse) → {boolean}**

**Get** or **Set** the reverse option value.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| newReverse | boolean | <optional> | The new reverse setting of the scene. |

Source:

* [ScrollMagic/Scene/getters-setters.js](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html), [line 203](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html#sunlight-1-line-203)

##### Fires:

* Scene.change,event: when used as setter

##### Returns:

* get - Current reverse option value.{ boolean }
* set - Parent object for chaining.{ Scene }

##### Example

// get the current reverse option  
var reverse = scene.reverse();  
  
// set new reverse option  
scene.reverse(false);

#### triggerElement**(newTriggerElement) → {string|object}**

**Get** or **Set** the triggerElement option value.

Does **not** fire Scene.shift, because changing the trigger Element doesn't necessarily mean the start position changes. This will be determined in Scene.refresh(), which is automatically triggered.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| newTriggerElement | string | object | <optional> | The new trigger element for the scene. |

Source:

* [ScrollMagic/Scene/getters-setters.js](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html), [line 165](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html#sunlight-1-line-165)

##### Fires:

* Scene.change,event: when used as setter

##### Returns:

* get - Current triggerElement.{ string | object }
* set - Parent object for chaining.{ Scene }

##### Example

// get the current triggerElement  
var triggerElement = scene.triggerElement();  
  
// set a new triggerElement using a selector  
scene.triggerElement("#trigger");  
// set a new triggerElement using a DOM object  
scene.triggerElement(document.getElementById("trigger"));

#### triggerHook**(newTriggerHook) → {number}**

**Get** or **Set** the triggerHook option value.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| newTriggerHook | number | string | <optional> | The new triggerHook of the scene. See Scene parameter description for value options. |

Source:

* [ScrollMagic/Scene/getters-setters.js](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html), [line 184](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html#sunlight-1-line-184)

##### Fires:

* Scene.change,event: when used as setter
* Scene.shift,event: when used as setter

##### Returns:

* get - Current triggerHook (ALWAYS numerical).{ number }
* set - Parent object for chaining.{ Scene }

##### Example

// get the current triggerHook value  
var triggerHook = scene.triggerHook();  
  
// set a new triggerHook using a string  
scene.triggerHook("onLeave");  
// set a new triggerHook using a number  
scene.triggerHook(0.7);

### Scene Properties (getter)

#### scrollOffset**() → {number}**

**Get** the current scroll offset for the start of the scene.

Mind, that the scrollOffset is related to the size of the container, if triggerHook is bigger than 0 (or "onLeave").

This means, that resizing the container or changing the triggerHook will influence the scene's start offset.

Source:

* [ScrollMagic/Scene/getters-setters.js](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html), [line 261](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html#sunlight-1-line-261)

##### Returns:

The scroll offset (of the container) at which the scene will trigger. Y value for vertical and X value for horizontal scrolls.

{ number }

##### Example

// get the current scroll offset for the start and end of the scene.  
var start = scene.scrollOffset();  
var end = scene.scrollOffset() + scene.duration();  
console.log("the scene starts at", start, "and ends at", end);

#### state**() → {string}**

**Get** the current state.

Source:

* [ScrollMagic/Scene/getters-setters.js](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html), [line 248](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html#sunlight-1-line-248)

##### Returns:

"BEFORE", "DURING" or "AFTER"

{ string }

##### Example

// get the current state  
var state = scene.state();

#### triggerPosition**() → {number}**

**Get** the trigger position of the scene (including the value of the offset option).

Source:

* [ScrollMagic/Scene/getters-setters.js](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html), [line 278](http://docs.google.com/ScrollMagic_Scene_getters-setters.js.html#sunlight-1-line-278)

##### Returns:

Start position of the scene. Top position value for vertical and left position value for horizontal scrolls.

{ number }

##### Example

// get the scene's trigger position  
var triggerPosition = scene.triggerPosition();

### Event Handling

#### off**(names, callback) → {Scene}**

Remove one or more event listener.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| names | string |  | The name or names of the event that should be removed. |
| callback | function | <optional> | A specific callback function that should be removed. If none is passed all callbacks to the event listener will be removed. |

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 268](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-268)

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

function callback (event) {  
 console.log("Event fired! (" + event.type + ")");  
}  
// add listeners  
scene.on("change update", callback);  
// remove listeners  
scene.off("change update", callback);

#### on**(names, callback) → {Scene}**

Add one ore more event listener.

The callback function will be fired at the respective event, and an object containing relevant data will be passed to the callback.

##### Parameters:

| Name | Type | Description |
| --- | --- | --- |
| names | string | The name or names of the event the callback should be attached to. |
| callback | function | A function that should be executed, when the event is dispatched. An event object will be passed to the callback. |

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 228](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-228)

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

function callback (event) {  
 console.log("Event fired! (" + event.type + ")");  
}  
// add listeners  
scene.on("change update progress start end enter leave", callback);

#### trigger**(name, vars) → {Scene}**

Trigger an event.

##### Parameters:

| Name | Type | Argument | Description |
| --- | --- | --- | --- |
| name | string |  | The name of the event that should be triggered. |
| vars | object | <optional> | An object containing info that should be passed to the callback. |

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 315](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-315)

##### Returns:

Parent object for chaining.

{ Scene }

##### Example

this.trigger("change");

### Events

#### add

Scene add event.

Fires when the scene is added to a controller.

This is mostly used by plugins to know that change might be due.

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | | controller | boolean | The controller object the scene was added to. | |

Since:

* 2.0.0

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 192](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-192)

##### Example

scene.on("add", function (event) {  
 console.log('Scene was added to a new controller.');  
});

#### change

Scene change event.

Fires whenvever a property of the scene is changed.

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | | what | string | Indicates what value has been changed | | newval | mixed | The new value of the changed property | |

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 130](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-130)

##### Example

scene.on("change", function (event) {  
 console.log("Scene Property \"" + event.what + "\" changed to " + event.newval);  
});

#### destroy

Scene destroy event.

Fires whenvever the scene is destroyed.

This can be used to tidy up custom behaviour used in events.

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | | reset | boolean | Indicates if the destroy method was called with reset true or false. | |

Since:

* 1.1.0

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 167](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-167)

##### Example

scene.on("enter", function (event) {  
 // add custom action  
 $("#my-elem").left("200");  
 })  
 .on("destroy", function (event) {  
 // reset my element to start position  
 if (event.reset) {  
 $("#my-elem").left("0");  
 }  
 });

#### end

Scene end event.

Fires whenever the scroll position its the ending point of the scene.

It will also fire when scrolling back up from after the scene and going over its end position. If you want something to happen only when scrolling down/right, use the scrollDirection parameter passed to the callback.

For details on this event and the order in which it is fired, please review the Scene.progress method.

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | | progress | number | Reflects the current progress of the scene | | state | string | The current state of the scene "DURING" or "AFTER" | | scrollDirection | string | Indicates which way we are scrolling "PAUSED", "FORWARD" or "REVERSE" | |

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 29](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-29)

##### Example

scene.on("end", function (event) {  
 console.log("Hit end point of scene.");  
});

#### enter

Scene enter event.

Fires whenever the scene enters the "DURING" state.

Keep in mind that it doesn't matter if the scene plays forward or backward: This event always fires when the scene enters its active scroll timeframe, regardless of the scroll-direction.

For details on this event and the order in which it is fired, please review the Scene.progress method.

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | | progress | number | Reflects the current progress of the scene | | state | string | The current state of the scene - always "DURING" | | scrollDirection | string | Indicates which way we are scrolling "PAUSED", "FORWARD" or "REVERSE" | |

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 50](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-50)

##### Example

scene.on("enter", function (event) {  
 console.log("Scene entered.");  
});

#### leave

Scene leave event.

Fires whenever the scene's state goes from "DURING" to either "BEFORE" or "AFTER".

Keep in mind that it doesn't matter if the scene plays forward or backward: This event always fires when the scene leaves its active scroll timeframe, regardless of the scroll-direction.

For details on this event and the order in which it is fired, please review the Scene.progress method.

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | | progress | number | Reflects the current progress of the scene | | state | string | The current state of the scene "BEFORE" or "AFTER" | | scrollDirection | string | Indicates which way we are scrolling "PAUSED", "FORWARD" or "REVERSE" | |

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 71](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-71)

##### Example

scene.on("leave", function (event) {  
 console.log("Scene left.");  
});

#### progress

Scene progress event.

Fires whenever the progress of the scene changes.

For details on this event and the order in which it is fired, please review the Scene.progress method.

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | | progress | number | Reflects the current progress of the scene | | state | string | The current state of the scene "BEFORE", "DURING" or "AFTER" | | scrollDirection | string | Indicates which way we are scrolling "PAUSED", "FORWARD" or "REVERSE" | |

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 110](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-110)

##### Example

scene.on("progress", function (event) {  
 console.log("Scene progress changed to " + event.progress);  
});

#### remove

Scene remove event.

Fires when the scene is removed from a controller.

This is mostly used by plugins to know that change might be due.

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | |

Since:

* 2.0.0

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 210](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-210)

##### Example

scene.on("remove", function (event) {  
 console.log('Scene was removed from its controller.');  
});

#### shift

Scene shift event.

Fires whenvever the start or end **scroll offset** of the scene change.

This happens explicitely, when one of these values change: offset, duration or triggerHook.

It will fire implicitly when the triggerElement changes, if the new element has a different position (most cases).

It will also fire implicitly when the size of the container changes and the triggerHook is anything other than onLeave.

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | | reason | string | Indicates why the scene has shifted | |

Since:

* 1.1.0

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 147](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-147)

##### Example

scene.on("shift", function (event) {  
 console.log("Scene moved, because the " + event.reason + " has changed.)");  
});

#### start

Scene start event.

Fires whenever the scroll position its the starting point of the scene.

It will also fire when scrolling back up going over the start position of the scene. If you want something to happen only when scrolling down/right, use the scrollDirection parameter passed to the callback.

For details on this event and the order in which it is fired, please review the Scene.progress method.

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | | progress | number | Reflects the current progress of the scene | | state | string | The current state of the scene "BEFORE" or "DURING" | | scrollDirection | string | Indicates which way we are scrolling "PAUSED", "FORWARD" or "REVERSE" | |

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 8](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-8)

##### Example

scene.on("start", function (event) {  
 console.log("Hit start point of scene.");  
});

#### update

Scene update event.

Fires whenever the scene is updated (but not necessarily changes the progress).

##### Properties:

| Name | Type | Description |
| --- | --- | --- |
| event | object | The event Object passed to each callback Properties  | Name | Type | Description | | --- | --- | --- | | type | string | The name of the event | | target | Scene | The Scene object that triggered this event | | startPos | number | The starting position of the scene (in relation to the conainer) | | endPos | number | The ending position of the scene (in relation to the conainer) | | scrollPos | number | The current scroll position of the container | |

Source:

* [ScrollMagic/Scene/event-management.js](http://docs.google.com/ScrollMagic_Scene_event-management.js.html), [line 92](http://docs.google.com/ScrollMagic_Scene_event-management.js.html#sunlight-1-line-92)

##### Example

scene.on("update", function (event) {  
 console.log("Scene updated.");  
});

© Jan Paepke 2015

Documentation generated by [JSDoc 3.5.5](https://github.com/jsdoc3/jsdoc) using a customized version of the [DocStrap template](https://github.com/terryweiss/docstrap).